1. 上课约定须知

课程主题：SparkCore 第二次课：SparkCore源码分析-- RPC 和 集群启动
上课时间：20:00 - 23:00
课件休息：21:30 左右 休息10分钟
课前签到：如果能听见音乐，能看到画面，请在直播间扣 666 签到

2. 上次作业复盘

使用 Spark 的 RPC 框架实现一款 C/S 架构的聊天应用程序。

要求：

```
1、该应用应用程序，有最基本的服务端程序和客户端程序
2、首先启动服务端，保证服务端一直运行
3、然后启动客户端，客户端向服务端注册
4、再启动其他多个客户端，待启动的客户端注册之后，和其他客户端（当然，当前客户端必须具备感知其他客户端存在的功能，然后选择通信对象）进行通信。
5、客户端关闭，其他客户端收到通知。
```

3. 上次内容总结

Spark 的学习，主要分为两个方面，一方面是 架构设计，一方面是源码解析。总共四次课

上节课，是 Spark 的第一节课，主要讲解的是 Spark 的架构设计，重点分析，Spark 有哪些工作组件，核心工作机制等。

```
1、Spark DAG 引擎剖析
2、Spark 核心功能和架构设计
3、Spark 运行机制和任务执行流程详解
4、Spark Shuffle详解
5、Spark 内存模型
```

从今天开始，进入到 Spark 的源码分析。

4. 本次内容大纲

从这次课开始，讲解的内容为 SparkCore 的源码分析，今天是第一次，重点讲解 Spark Standalone 集群启动，内容大纲：

```
1、Spark 网络通信 RPC 实现
2、Spark 集群启动脚本 start-all.sh 源码分析
3、Spark 主节点 Master 启动源码分析
4、Spark 从节点 Worker 启动源码分析
```

5. 详细课堂内容

5.1. Spark 网络通信 RPC 实现

Spark 的 RPC 到现在为止，有两种大版本的实现：

```
1、Akka 库 scala Spark-1.x 网络通信机制的具体实现
2、Netty 库 java Spark-2.x 网络通信机制的具体实现
```

详细的两种版本的RPC见文档！

1、基于 Akka 实现网络通信
2、基于 Netty 实现网络通信

5.2. Spark 集群启动脚本 start-all.sh 源码分析

在 Spark 源码项目中，有一个 bin 目录，也有一个 sbin 目录，集群启动的脚本，位于 sbin 目录中。

启动命令：

```
start-all.sh
```

内部执行：

```
start-master.sh
start-slaves.sh  --- start-slave.sh
```

都共同调用：

```
spark-daemon.sh
```

内部调用：

```
spark-class
```

shell 内部会调用 org.apache.spark.launcher.Main 帮忙整理运行参数，组装启动 shell 命令，最终转到执行：

```
Master启动：java org.apache.spark.deploy.master.Master
Worker启动：java org.apache.sprak.deploy.worker.Worker
```

5.3. Spark 主节点 Master 启动源码分析

根据以上的脚本分析：

```
org.apache.spark.deploy.master.Master.main()
```

三个重要的代码入口：

```
1、Master 的 Main 方法
    rpcEnv.setupEndpoint(master_name, new Master(....))
2、Master 的 构造方法
3、Master 的 onStart() 方法
```

干的四件重要的事情：

```
1、启动 rpc 服务端
2、启动 web ui
3、选举 active master
4、启动了一个定时任务：每隔一段时间去检查 dead workers
```

5.4. Spark 从节点 Worker 启动源码分析

根据以上的脚本分析：

```
org.apache.sprak.deploy.worker.Worker.main()
```

干的两件重要的事情：

```
1、启动 RPC 服务
2、启动 web ui
3、先注册
4、注册成功之后，定时发心跳
```

worker已经启动正常。！

6. 本次课程总结

今天的主要内容：

```
1、Spark RPC
2、Spark 集群启动
```

7. 本次课程作业

使用 Spark 的 RPC 框架实现一款 C/S 架构的聊天应用程序。

要求：

```
1、该应用应用程序，有最基本的服务端程序和客户端程序
2、首先启动服务端，保证服务端一直运行
3、然后启动客户端，客户端向服务端注册
4、再启动其他多个客户端，待启动的客户端注册之后，和其他客户端（当然，当前客户端必须具备感知其他客户端存在的功能，然后选择通信对象）进行通信。
5、客户端关闭，其他客户端收到通知。
```